Setup for Mach 3.0

Mary R. Thompson

1 April 1993

1. Introduction

Booting a micro-kernel/server system is slightly more complicated than booting a macro-kernel system since more files have to be in place and functioning correctly before the system will accept input from the user. The CMU environment has also contributed some complexity to the usual BSD Unix setup.

In Mach 3.0 the pieces that are needed for a sucessful boot are: the 16-sector boot code (usually supplied by the machine vendor), the micro-kernel, a paging file, a server for the kernel to call, an emulation library (at least for our servers) and a startup user program.

The other complications are the fact that we normally have a super root and local root in order to support the CMU RFS file system and the fact that when you come up single user, you do not have root access, but are running with the userid "opr". Also on CMU machines we have an /etc/rc that insists that /vmunix be a symbolic link to the unix-server you are running before it will complete a boot to multi-user.

This document explains exactly what is required to boot a Mach 3.0 system up multi-user. It explains the non-Uxix features, how to set up your machine originally, how to safely boot alternative pieces of the OS and how external sites can get the OS files.

2. Super root and local root

Mach systems (2.5 or 3.0) are normally set up with a super root that contains one real directory ***RFS***. ***RFS*** contains the directory ***.LOCALROOT*** and links to all other machines that are accessible via RFS. The super root also contains symbolic links to all the standard directories in /, eg. ***/dev, /mnt, /bin, /etc, /lib, /usr, /usr1***. ***/RFS/.LOCALROOT*** is known as the local root. After boot time the local root is named / and the super root is named */../../.* Some files such as the Mach kernel and the Unix-server have hard links in both the local root and super root, so that they can be found by the same name during and after the boot sequence. In the Mach 3.0 case the switching of the root, /, from the super root to the local root is done by the Unix-server or POE rather than the micro-kernel. If you are not using the RFS filesystem you can use only a single root by not having the /RFS directory. You may see a complaint during the boot sequence about i(unable to change to local root), but the system will work ok.

3. Setup for a default boot

3.1. Micro-kernel Names

In Mach 2.5 the kernel is called both ***mach*** and ***vmunix*** and is hard-linked to by ***/../../vmunix, /../../mach, /mach*** and ***/vmunix***. Some boot code looks for the file ***/vmunix*** by default and some looks for ***/mach*** but always on the super root. As the kernel boots, the root is changed to be the local root, ***/RFS/.LOCALROOT***. After booting some Unix programs read the "Unix namelist" by referencing ***/vmunix***.

In Mach 3.0 the names ***mach*** and ***mach.boot*** are used for the Mach micro-kernel and ***vmunix*** and ***startup*** are used for the Unix-server. The micro-kernel is the program that you wish to boot. Boot code that we provide for the i386 will boot ***mach*** on device 0, partition a by default. Older boot code for the Vax and Sun3's may boot ***vmunix*** by default. On the DecStations, the default boot name is setable. On DecStations setup by CMUCS facilities the default is ***mach_kernel*** on device 0, partition 0. Boot programs normally allow you to specify a non-default name for the program to be booted, so you can either install the micro kernel on your machine by whatever default name your boot code uses, or specify the boot file by name whenever you boot the machine. The Mach micro-kernel is named ***mach.boot.MKnn.<config>*** in the build and release directories. Note that in releases prior to MK68 the kernel was named ***mach_kernel.MKnn.<config>*** but now the file by that name is not a bootable file. The Unix-server is named ***vmunix.UKnn.<config>*** in the build and release directories.

3.2. Other file names

The ***/mach_servers*** directory is the default directory where the Mach 3.0 kernel and various servers look for files. There are four required file names that are coded into the micro-kernel and unix-servers and cannot be changed at boot time. They are ***paging_file, startup, emulator*** and ***mach_init***. The name ***mach_servers*** may be specified at boot time to allow the booting of alternative pieces of the OS, but in this section the default boot is assumed.

/mach_servers may be a subdirectory in either the local root or the super root with a symbolic link to it in the other root or there can be two different directories. A directory of this name must be findable from both the root and super-root and at least one of the directories must be on the root partition. If you do not have enough room on your root partition for all the files that are required to boot, the mach_servers directory on the superoot can be a symbolic link to a directory on a different partition. This link is read by the micro-kernel and must be of the form */dev/<partition>/mach_servers*, by which the micro-kernel reads unmounted partitions.

The micro-kernel by default looks in the ***/mach_servers*** directory on the super root for three files: ***startup*** which is the program it will start the user task in; ***emulator*** which the kernel reads to initialize the symbols for the kernel debugger; and ***paging_file*** which is the default paging space. These names may refer to files in that directory, symbolic links to files on the root partition or symbolic links to files on other *unmounted* partitions. See the next section for an example of this last case.

The Unix-server by default looks in the ***/mach_servers*** directory on the local root for two files: ***emulator*** and ***mach_init***. ***emulator*** is the emulation library that is loaded with each task that the Unix-server starts. ***mach_init*** is the program called by the first task started by the Unix-server. These files must reside on the root partition, since the Unix-server only has the root partition mounted and does not read unmounted partitions like the micro-kernel did.

If you are running POE rather than the Unix-server it looks in the ***/mach_servers*** directory on the local root for two files: ***poe_emulator*** and ***poe_init***. ***poe_emulator*** is the emulation library that is loaded with each task that POE starts. ***poe_init*** is the program that is run by the the first task POE starts. As in the case of the Unix server, these programs must reside on the root partition.

3.3. Installing the pieces

First you need to install a Mach 3.0 micro-kernel on the root. Copy a "mach.boot" to ***/mach*** and create a hard-link to it from ***/../../mach*** or whatever your default boot file name is. ***mach.boot*** is built from the sources in the mach3.kernel collection or can be found in mach3.release in the directory ***special.*** It's name in those directories is of the form ***mach.boot.MKnn.<congig>***.

Now you need to make a ***/mach_servers*** directory, the recommended way is:

```
mkdir /../../mach_servers
cd /
ln -s ../../mach_servers mach_servers
```

The alternative method to conserve space on the root partition (assuming that /dev/sd0e is mounted on /usr) is to:

```
mkdir /mach_servers
mkdir /usr/mach_servers
cd /../..
ln -s /dev/sd0e/mach_servers mach_servers
```

This second alternative is the default setup for DecStations at CMU, so be sure you know what you are doing if you try to install new pieces of the OS on one of those machines.

Now populate ***/mach_servers*** with:

paging_file this is usually a symbolic link to a file on an umounted partition in order to avoid using large amounts of space on the root partition. For example if ***/usr1*** is mounted on ***/dev/hd0f*** and has enough free space you could make ***paging_file*** be a symbolic link to "/dev/hd0f/pagingfile". Then you create a ***/usr1/pagingfile*** of sufficient size to handle your paging needs. Between about 4-20M is recommended. The system does not grow this file, so you must set it up in advance as a big file. An easy way to pre-assign the space is to do:

```
dd if=/dev/rhd0f of=/usr1/pagingfile bs=1024k count=n
```

which creates a nMeg file. It is possible to have Mach 2.5 and 3.0 use the same paging file, but care must be taken so that Mach 2.5 does not truncate the file when it boots. To ensure this the line in ***/etc/fstab*** that specifies the paging file for Mach 2.5 should have the value of pagelowat equal to that of pagehiwat. The system will boot without a paging files and run until it needs to page something out. Just answer the request during the boot process for a paging file with a carriage return. This may be useful if you are booting off a floppy or are just trying to come up single-user to fix something.

startup the program that the micro-kernel calls. This file must be found in the /mach_servers directory that is on (or linked from) the super-root. It does not need to physically reside on the root partition. This is the way putting the "super-root/mach_servers" directory off the root partition saves space on the root partition. Usually this is the Unix-server or POE. It could be an OS program of your own. The name of the unix server is ***vmunix.UXnn.<config>*** and is built from the sources in the mach3.unix release or can be found in the mach3.release collection in the directory ***special.*** The name of POE is ***poe.POEn***. It is built from the sources in mach3.poe or can be found in the mach3.release collections in the directory ***special.***

emulator the emulation library which the Unix-server loads with each task and the micro-kernel reads symbols from. This file must be accessible both from the super-root where the micro-kernel reads and from the local root where the Unix-server loads it. If you are using separate mach_servers directories you either need two copies of this file, or one copy on the root partition and a symlink that that the micro-kernel can follow from the non-root directory. This program is built from the sources in the mach3.unix collection or can be found in the mach3.release collection as ***special/emulator.UXnn***.

mach_init	the program that is run by the first task the Unix-server creates. It must be installed in ***/mach_servers*** on the local root. It is built from sources in the mach3.user collection in the directory ***etc/mach_init*** or can be found in mach3.release ***etc/mach_init***. THIS PROGRAM IS NOT COMPATIBLE WITH THE MACH 2.5 VERSION. You may set ***/mach_servers/mach_init*** to be a symbolic link to the Mach 2.5 version of ***/etc/init***. In this case the system will boot, but Mach servers will not work.
poe_init	the program that is run by the first task POE creates. It should be installed as ***/mach_servers/poe_init*** if you are going to run POE. It provides a user shell for a single user under POE. It is built from the sources in the mach3.poe collection or can be found in mach3.release as ***etc/poe_init***.
poe_emulator	the emulation library which POE loads with each task. It is built from the sources in the mach3.poe collection or can be found in mach3.release as ***special/poe_emulator.POEn***. It should be installed as ***/mach_servers/poe_emulator*** if you are going to run POE.

4. Booting new kernel/server pieces

If you wish to try out a new version of any of the critical OS pieces you should be sure that you have a system that is known to work in a place where it can be found by the boot process. The bootcode, the micro-kernel and the unix-server allow you to specify an alternative name for the micro-kernel and the ***/mach_servers*** directory. You cannot specify the names of specific files that live in the </mach_servers> directory, so even if you want to change only one file there, you must create a fully populated directory with that file and copies or links of the old files in it.

Only the boot code needs to know the name and location of the micro-kernel, so how you specify that is dependent on what boot code you are running. The boot code we use accepts several options to be passed to the booted program (the micro-kernel). We use *-s* for single-user boot and *-q* (DecStation) or *-a* (Sun3,Vax and I386) to prompt for an alternative mach_servers directory. Since the only information the micro-kernel and unix-server have is an alternative directory name, this directory must contain a complete set of the four files needed for booting.

Some examples of how to set things up follow:

Booting a new micro-kernel, standard unix-server et.al

install new kernel as mach.new on superroot.
do an /etc/halt or /etc/shutdown

at boot prompt type:

DecStation 5000:	>>**boot 3/rz0/mach.new**
i386:	boot **mach.new or sd(0,a)mach.new**
Sun3:	>>**boot mach.new or boot sd(0,0,0)mach.new**
Micro Vax:	>>**b/3 mach.new**

Booting a new unix-server and emultor, standard micro-kernel

Create mach_servers.new directories or links on root and super-root

If you only have one directory (and one link) populate it with

```
      startup  -- new unix-server
      emulator -- new emulator code
      paging_file  -- link to existing pagingfile
      mach_init -- link to existing version of mach_init.
```

If you have two directories populate /../../mach_servers.new with

```
      startup  -- new unix-server
      emulator -- new emulator code (may be a symlink)
      paging_file  -- link to existing pagingfile
```

and /mach_servers.new with

```
      emulator  -- new emulator code
      mach_init -- link to existing version of mach_init
```

do and /etc/halt or /etc/shutdown

at the boot prompt type:

```
   DecStation 5000:   >>boot -sq
   i386:              boot -sa
   Sun3:              >>boot -sa
   Micro Vax:         >>b/3 -sa
```

The the **-s** switch will cause the kernel to drop into the debugger as soon as is possible to allow you to set any break points. Typing c causes the boot to be continued. It also causes the system to come up single-user. You will prompted for the name of the root device and the /mach_servers directory.

Booting both a new micro-kernel and unix-server

With the setup as in the previous examples respond to the boot prompt

```
   DecStation 5000:   >>boot 3/rz0/mach.new -sq
   i386:              boot mach.new -sa
   Sun3:              >>boot sd(0,0,0)mach.new -sa
   Micro Vax:         >>b/3 mach.new -sa
```

5. Mach 3.0 Servers

If it exists ***/mach_servers/mach_init*** is the first program called by the Unix-server. If it does not exist ***/etc/init*** is called and your system will boot but none of the mach servers will work. ***mach_init*** initializes the following ports: **name_server_port, environment_port** and **service_port**. The first port is used by either the NetMsgServer or snames depending on which you choose to run. The second is used by the Environment Manager and the third by mach_init itself as the Service server. All the processes on the system inherit these ports.

The collection mach3.release contains executables of mach3 programs and servers. They can be installed anywhere on your system and your PATH variable set to find them.

It is assumed below that you have copied the contents of the mach3.release collection (except for the directory ***special***) to ***/usr/mach3***. There are two servers that we normally run:

/usr/mach3/bin/snames

provides a local name service that supplies the same name lookup functions (on the same port) that the NetMsgServer does. Snames' name space is local to the machine it is running on while the NetMsgServer's name space spans all machines that are running instantiations of the NetMsgServer. Only one of these servers can be run.

/usr/mach3/bin/machid
provides the machid services which are needed by such programs as ms, pinfo, vmstat and gdb4.5. Machid needs to have a name server running.

6. Accessing AFS

Since UX29 Mach 3.0 has used the same AFS file access code as Mach 2.5. Thus both systems look in /usr/vice/etc/cacheinfo to find the location of the venus.cache directory. Note: that the name given in cacheinfo should be a symbolic link to the real venus cache.

The daemon afsd (or afsd.31) must be run to get AFS access. In Mach 2.5 this is started by nanny with the script /usr/misc/.afs/etc/launchafsd. This will continue to work in Mach 3.0.

7. Finding the files

7.1. Sup Collections

Mach 3.0 is suped as 6 separate collections. The collections mach3.kernel, mach3.unix, mach3.poe, mach3.user and mach3.buildtools contain only source files. When you build these sources directories named obj/<machine_type> will be created. These directories are referred to as *build directories.* In the past you have been expected to copy the programs that you built from these directories to the places from which they were used. Starting with releases MK69, UX31, POE9 and USER12 the build process copies the end products of a build to the *release directory*, release/<machine_type>. The major servers and the kernel files are put into the directory release/<machine_type>/special, from which they must be copied to the directories in which they are used. Other programs and libraries are put into release/<machine_type>/{bin,include,etc,lib}. These directories are normally added to the approriate path names when a Mach 3.0 system is running so the files can be used directly from there.

The sixth Mach3 collection, Mach3.release, provides a copy of the latest binaries that were built at CMU. You may use these to facilitate boot-strapping or as a binary system if you do not want to make any changes and trust other people's binaries.

7.2. AFS release directories

If you are at CMU or have AFS access to cs.cmu.edu, the kernel and ux binary files can be found in /afs/cs.cmu.edu/project/mach3/latest/release/@sys/special and mach_init can be found in .../release/@sys/etc/mach_init.